

# On the Capacity Requirement for Arbitrary End-to-End Deadline and Reliability Guarantees in Multi-hop Networks

Han Deng

Department of ECE  
Texas A&M University  
College Station, TX 77840, USA  
Email: hdeng@email.tamu.edu

I-Hong Hou

Department of ECE  
Texas A&M University  
College Station, TX 77840, USA  
Email: ihou@tamu.edu

**Abstract**—It has been shown that it is impossible to achieve both stringent end-to-end deadline and reliability guarantees in a large network without having complete information of all future packet arrivals. In order to maintain desirable performance in the presence of uncertainty of future packet arrivals, common practice is to add redundancy by increasing link capacities. This paper studies the amount of capacity needed to provide stringent performance guarantees. We propose a low-complexity online algorithm and prove that it only requires a small amount of redundancy to guarantee both end-to-end deadline and reliability. Further, we show that in large networks with very high reliability requirements, the redundancy needed by our policy is at most twice as large as a theoretical lower bound. Also, for practical implementation, we propose a fully distributed protocol based on the previous centralized policy. Without adding redundancy, we further propose a low-complexity order-optimal online policy for the network. Simulation results also show that our policy achieves much better performance than other state-of-the-art policies.

## I. INTRODUCTION

Many emerging safety-critical applications, such as Internet of Things (IoT) and Cyber-Physical Systems (CPS), require communication protocols that support strict end-to-end delay and reliability guarantees for all packets. In a typical scenario, when sensors detect unusual events that can cause system instability, they send out this information to actuators or control centers. This information needs to be delivered within a strict deadline for actuators or control centers to resolve the unusual events. The system can suffer from a critical fault when a small portion of packets fail to be delivered on time.

Despite the huge literature on quality of service (QoS), there is little work that can provide end-to-end delay and reliability guarantees simultaneously, especially when packet arrivals are time-varying and unpredictable. The lack of progress is mainly caused by two fundamental challenges. On one hand, it is obvious that one cannot design the optimal network policies without obtaining complete knowledge of future packet arrivals and incurring high computation complexity. Therefore, practical so-

lutions need to rely on online suboptimal policies. On the other hand, in a multi-hop network, the scheduling decision of one communication link will impact the decisions of subsequent links. The negative effects of suboptimal decisions by online policies therefore get accumulated along the path of multi-hop transmissions. In fact, a recent work by Mao, Koksai, and Shroff [1] has proved that the performance of any online policies deteriorates as the length of the longest path in the network increases. As a result, no online policy can provide meaningful performance guarantees when the size of the network is large.

In order to maintain desirable performance using online suboptimal policies, current practice is to add redundancy into the system. During system deployment, the capacities of communication links are chosen to be larger than necessary. Such redundancy alleviates the negative impacts of suboptimal decisions by online policies. Using this approach, a critical question is to determine the amount of redundancy needed to provide the desirable performance guarantees. This paper aims to answer this question.

We first show that the problem of maximizing the number of timely packet deliveries can be formulated as a linear programming problem when one knows the complete knowledge of all future packet arrivals. In the setting of online policies, some of the parameters of this linear programming problem will only be revealed when the corresponding packets arrive. Therefore, online policies need to make routing and scheduling decisions for packets without knowing all parameters. On the other hand, we also observe that adding redundancy by increasing link capacities is equivalent to relaxing a subset of constraints in the linear programming problem. Based on these observations, we define a competitive ratio that, given the amount of redundancy, quantifies the relative performance of online policies in comparison to the optimal offline solution.

Using the primal-dual method, we propose an online policy that achieves good performance in terms of com-

petitive ratio. This policy has several important features: First, when there is no redundancy added to the system, the performance of our online policy is asymptotically better than that of the recent work [1] when the size of the network increases. Second, we also show that only a small amount of redundancy is needed to achieve strict performance guarantees. Specifically, in order to guarantee the timely delivery of at least  $1 - \frac{1}{\theta}$  as many packets as the optimal solution in a network whose longest path has length  $L$ , our policy only needs to increase link capacities by  $\ln L + \ln \theta$  times. Finally, we also show that our policy can be implemented with very low complexity.

Next, we establish a theoretical lower bound of competitive ratio for all online policies. We show that, in order to guarantee a certain degree of performance, the redundancy needed by our policy is only a small amount away from the theoretical limit. In particular, when both  $L$  and  $\theta$ , as defined in the previous paragraph, go to infinity, the redundancy needed by our policy is at most twice as large as the theoretical limit.

We also study online policies when one cannot increase network capacity by adding redundancy. We propose another online policy and prove that it is order optimal with fixed link capacity. Specifically, we show that this online policy guarantees to deliver at least  $\frac{1}{O(\log L)}$  as many packets before their deadlines as the optimal offline solution, where  $L$  is the maximum route length. As the previous study [1] has proved no online policy can deliver more than  $\frac{1}{O(\log L)}$  packets without redundancy, our policy is order-optimal.

While neither of our online policies need any information about future packet arrivals to make routing and scheduling decisions, they are centralized algorithms that require tight coordination. For large networks without a centralized coordinator, we also propose a fully distributed protocol that is inspired by the design principles of our centralized online policies. This distributed protocol only requires each node to broadcast its local congestion information very infrequently, and therefore it only incurs a small amount of communication overhead. When a packet arrives at a source node, the source node determines a suggested route for the packet using its received congestion information, and each link on the route makes scheduling decisions solely based on its local information.

All three of our policies are evaluated by simulations. We compare our policies with the widely used earliest deadline first policy (EDF) and recent policy studied in [1]. Simulation results show that all our policies perform better than the other two policies. This result is in particular surprising because our distributed protocol even achieves better performance than the online policy in [1], which is a centralized one.

The rest of the paper is organized as follows. Section II reviews some existing works. Section III introduces our system model defines the competitive ratio. Section IV

proposes our online policy and studies its competitive ratio and computation complexity. Section V establishes a theoretical lower bound of competitive ratio. Section VI proposes an order-optimal policy and studies its competitive ratio. Section VII proposes a distributed protocol based on the intuitions of our centralized online policy. Section VIII provides simulation on our proposed algorithms and compare them with two other online policies. Finally, Section IX concludes this paper.

## II. RELATED WORK

Online scheduling problem in real-time environment has been studied in many previous works. Studies show that earliest deadline first algorithm (EDF) [2], [3] and least laxity first algorithm (LLF) [3] achieve the same performance as the optimal offline algorithm when the system is under-loaded, that is, the optimal offline algorithm can serve all jobs in the system. In under-loaded system, all jobs enter the system can be served by EDF and we do not need to drop any job when it arrives at the system. However, in over-loaded system, even with optimal offline algorithm, there are still some jobs that cannot be served. EDF and LLF achieve the same performance as the optimal online policy when the system is over-loaded. Also [3] proved that no online algorithm can guarantee to serve more than  $1/4$  of the jobs that can be served by optimal offline algorithm and provided an algorithm in a uniprocessor system which achieves  $1/4$  service bound. [4], [5] consider admission control in online scheduling. In [4], when all jobs have equal length, the competitive ratio of deterministic algorithm is bounded by 2. [6] considers the similar model as [4]. It introduces a parameter  $k$  to indicate the willingness of a job to have a delay before being served. It shows that when all jobs have equal length, the competitive ratio of deterministic algorithm is  $(1 + 1/(\lfloor k \rfloor + 1))$ -competitive instead.

In addition, online scheduling with multiple-server case has also been studied. [7] studies the scheduling of equal length jobs on two identical machines. [8]–[10] studies the case with parallel machines. The scheduler need to decide whether to accept or reject a packet and which machines is chosen to serve the job. [9] has proposed an algorithm with immediate decision which approaches  $\frac{e}{e-1}$ -competitive when the number of machines is greater or equal to 3. It also provide another lower bound that deterministic online algorithm with immediate decision is no better than 1.8-competitive when there are 2 machines. Later [10] has shown that online algorithm which makes immediate decision upon job releasing is bounded by  $\frac{e}{e-1}$ -competitive for multiple machine case.

There are also many works studying the scheduling problem in multihop network. An early study [11] focuses on the problem of packet scheduling with arbitrary end-to-end delay, fix route, and known packet injection rate. It propose a distributed algorithm which achieve a certain

delay bound. [12] studies the scheduling problem on a tree network. Packets arrive at an arbitrary node and they need to be transmitted to root node before the deadlines. Any packet that cannot arrive root node within deadline is considered lost. Thus this is also a fix route problem. The goal is to minimize the total lost packets. Shortest time to extinction (STE) algorithm is proposed and it is shown to achieve the performance of optimal offline policy. Also there are many works studying the end-to-end delay in multihop network. Rodoplu *et al.* [13] have studied the problem of dynamic estimating end-to-end delay over multi-hop mobile wireless networks. Sanada Komuro and Sekiya [14] have used Markov-chain model to study the string-topology multi-hop network and analyse the end-to-end throughput and delay. Jiao *et al.* [15] have studied the problem of estimating the end-to-end delay distribution for general traffic arrival model and Nakagami-m channel model by analyzing packet delay at each hop. Li *et al.* [16] ] have proposed using expected end-to-end delay for selecting path in wireless mesh networks. The expected end-to-end delay takes both queuing delay and delay caused by unsuccessful wireless transmissions. However, their work only aims at minimizing the average end-to-end delays, and cannot provide guarantees on per-packet delays.

Li and Eryilmaz [17] has studied the end-to-end deadline constrained traffic scheduling in multihop network. They develop algorithms to meet the deadline and throughput requirement in a wired network. However, they only consider the fix route model and they do not provide any performance guarantee. Wang *et al.* [18] have studied the problem of routing and scheduling on multi-hop wireless sensor network in order to optimize the system with the constraint of end-to-end delay and proposed a sub-optimal algorithm. Hou [19] proposed a throughput optimal policy for up-link tree networks with end-to-end delay constraints and delivery ratio requirement. The packets deadlines are the end of the frames in which they are generated. Singh and Kumar [20] have proposed a scheduling policy which maximize the throughput for multi-hop wireless networks. However, the paper uses a fix-route model and does not consider end-to-end delay. Mao, Koksall and Shroff [1] also considers a fix route problem. The network has arbitrary packet arrival and packet weight. The paper aims to maximize the total cumulative weight of packets that reach destination before their deadline. The paper has proved that the competitive ratio of any online policy is no better than  $O(\log L)$ , where  $L$  is the length of the maximum route. It has also proposed an admission control and packet scheduling policy and shown that it is  $O(L \log L)$ -competitive. Liu and Yang [21] have studied the multi-hop routing problem with hard end-to-end delay and the throughput region. They also assume that packets are required to be delivered to destination within one frame and the performance is evaluated by simulation. Our work will focus on online

routing and scheduling on multi-hop network with end-to-end delay constraint and aim to guarantee both packet deadline and network delivery ratio.

### III. SYSTEM MODEL

We consider a network with multihop transmissions. The network is represented by a directed graph where each node represents a router and an edge from one node to another represents a link between the corresponding routers. Packets arrive at their respective source nodes following some unknown sequence. We use  $\mathcal{M}$  to denote the set of all packets and  $\mathcal{L}$  the set of all links. When a packet  $m \in \mathcal{M}$  arrives at its source node, it specifies its destination and a deadline. The packet requests to be delivered to its destination before its specified deadline. Packets that are not delivered on time do not have any value, and can be dropped from the network. We aim to deliver as many packets on time as possible.

We assume that time is slotted and numbered by  $t = \{1, 2, 3, \dots\}$ . Different links in the network may have different link capacities, and we denote by  $C_l$  the number of packets that link  $l$  can transmit in a time slot. At the beginning of each time slot, each node decides which packets to transmit over its links, subject to capacity constraints of the links. Packets transmitted toward a node in time slot  $t$  will be received by that node at the end of the time slot, so that the node can transmit these packets to subsequent nodes starting from time slot  $t + 1$ .

Delivering a packet to its destination before its deadline require determining two things: the route used to forward the packet from its source to its destination, and the times at which the packet is transmitted along its route. We define a *valid schedule* for each packet  $m$  as the collection of links of a route, as well as the times of transmissions for each of these links, so that packet  $m$  can be delivered to its destination on time. For example, consider the network shown in Fig. 1. Suppose a packet arrives at node A at time slot 1, and needs to be delivered to node F before the end of time slot 3. One valid schedule for this packet is to transmit it over link d in time slot 1, and then over link g in time slot 2. We use  $\{(d, 1), (g, 2)\}$  to represent this valid schedule. Other valid schedules include  $\{(d, 1), (g, 3)\}$ ,  $\{(e, 1), (f, 2), (g, 3)\}$ , etc. On the other hand,  $\{(d, 1), (g, 4)\}$  is not a valid schedule because the packet is delivered to its destination after its deadline at time slot 4. The schedule  $\{(d, 3), (g, 2)\}$  is not valid because it would require node D to transmit the packet over link g at time slot 2 before it receives the packet at time slot 3. For each packet  $m$ , we let  $V(m)$  denote the set of valid schedules for  $m$ . The problem of deciding how to deliver packets on time then becomes one of choosing valid schedules for packets.

We use  $X_{mk}$  to denote the schedule selection for packet  $m$ . If  $X_{mk} = 1$ , packet  $m$  is transmitted using valid schedule  $k$ , and  $X_{mk} = 0$ , otherwise. Given the information of all packets, the problem of maximizing the total number

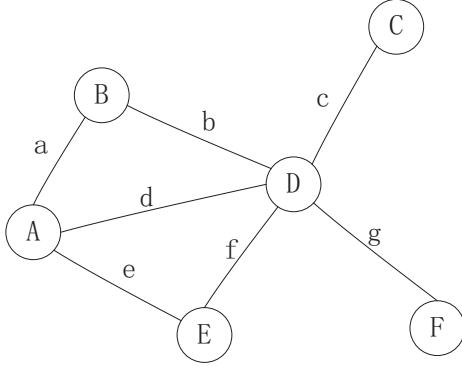


Fig. 1. Network topology.

of successful deliveries can be formulated as the following linear programming problem:

**Schedule:**

$$\text{Max} \sum_{m,k:k \in V(m)} X_{mk} \quad (1)$$

$$\text{s.t.} \sum_{k:k \in V(m)} X_{mk} \leq 1, \forall m \in \mathcal{M}, \quad (2)$$

$$\sum_{m,k:(l,t) \in k} X_{mk} \leq C_l, \forall l \in \mathcal{L}, t \in \{1, 2, \dots\}, \quad (3)$$

$$X_{mk} \geq 0, \forall m \in \mathcal{M}, k \in V(m). \quad (4)$$

Since  $X_{mk} = 1$  if packet  $m$  is transmitted using valid schedule  $k$ , Eq. (1) is the total number of packets that are delivered on time. Eq. (2) states that at most one valid schedule can be chosen for each packet. Eq. (3) states that each link can transmit at most  $C_l$  packets in any time slot. In practice,  $X_{mk}$  can only be either 0 or 1, but our problem formulation allows  $X_{mk}$  to be any real number in  $[0, 1]$ . Thus, the optimal solution to **Schedule** describes an upper bound on the total number of successful deliveries.

If information of all packets is available when the system starts, the optimal solution to **Schedule** can be found by standard linear programming methods. In practice, however, packets arrive sequentially, and we need to rely on online policies that determines the values of  $X_{mk}$  for each arriving packet  $m$  without knowing future packet arrivals. Without the knowledge of future arrivals, it is obvious that online policies cannot always achieve the optimal solution to **Schedule**. In fact, a recent work [1] has shown that, when the longest path between a source node and a destination node is  $L$ , no online policy can guarantee to deliver more than  $\frac{1}{\log_2 L}$  as many packets as the optimal solution. To put this number in perspective, consider a medium-sized network with  $L = 8$ . Even when the optimal solution can deliver all packets on time, the bound in the recent work states that no online policy can guarantee to deliver more than  $\frac{1}{\log_2 8} = \frac{1}{3}$  of all packets.

Such performance of online policies is unacceptable for virtually any applications.

In order to achieve good performance for online policies in the presence of unknown future arrivals, we consider the scenario where service providers can increase link capacities by, for example, upgrading network infrastructures. When the link capacities are increased by  $R$  times, link  $l$  can transmit  $RC_l$  packets in each time slot. With the increase in capacities, our problem can be rewritten as follows:

**Schedule( $R$ ):**

$$\text{Max} \sum_{m,k:k \in V(m)} X_{mk} \quad (5)$$

$$\text{s.t.} \sum_{k:k \in V(m)} X_{mk} \leq 1, \forall m \in \mathcal{M}, \quad (6)$$

$$\sum_{m,k:(l,t) \in k} X_{mk} \leq RC_l, \forall l \in \mathcal{L}, t \in \{1, 2, \dots\}, \quad (7)$$

$$X_{mk} \geq 0, \forall m \in \mathcal{M}, k \in V(m). \quad (8)$$

To evaluate the performance of online policies, we define a competitive ratio that incorporates the increase in capacities:

**Definition 1:** Given a sequence of packet arrivals, let  $\Gamma_{opt}$  be the optimal value of  $\sum_{m,k:k \in V(m)} X_{mk}$  in **Schedule**, and  $\Gamma_\eta(R)$  be the number of packets that are delivered under an online policy  $\eta$  when the link capacities are increased by  $R$  times. The online policy  $\eta$  is said to be  $(R, \rho)$ -competitive if  $\Gamma_{opt}/\Gamma_\eta(R) \leq \rho$ , for any sequence of packet arrivals.

#### IV. AN ONLINE ALGORITHM AND ITS COMPETITIVE RATIO

##### A. Algorithm Description

In this section, we propose an online policy based on primal-dual method and analyze the competitive ratio. We first note that the dual problem of **Schedule** is:

**Dual:**

$$\text{Min} \sum_m \alpha_m + \sum_{l,t} C_l \beta_{lt}, \quad (9)$$

$$\text{s.t.} \alpha_m + \sum_{l,t:(l,t) \in k} \beta_{lt} \geq 1, \forall m \in \mathcal{M}, k \in V(m) \quad (10)$$

$$\alpha_m \geq 0, \forall m, \quad (11)$$

$$\beta_{lt} \geq 0, \forall l, t, \quad (12)$$

where  $\alpha_m$  is the Lagrange multiplier corresponding to constraint (2), and  $\beta_{lt}$  is the Lagrange multiplier corresponding to constraint (3).

By the Weak Duality Theorem, we have the following lemma:

**Lemma 1:** Given any vectors of  $\{\alpha_m\}$  and  $\{\beta_{lt}\}$  that satisfy the constraints (10)–(12), we have  $\sum_m \alpha_m + \sum_{(l,t)} C_l \beta_{lt} \geq \Gamma_{opt}$ .

We now introduce our online algorithm. Our algorithm constructs  $\{X_{mk}\}, \{\alpha_m\}, \{\beta_{lt}\}$  simultaneously while ensuring they satisfy all constraints in **Schedule( $R$ )** and **Dual**. Initially, it sets  $\beta_{lt} \equiv 0$ . When a packet  $m$  arrives, the algorithm finds the valid schedule  $k^*$  that has the largest  $(1 - \sum_{l,t:(l,t) \in k} \beta_{lt})$  among all  $k \in V(m)$ . If  $1 - \sum_{l,t:(l,t) \in k^*} \beta_{lt} \leq 0$ , then the algorithm drops packet  $m$  and sets  $\alpha_m = 0$  and  $X_{mk} = 0$ , for all  $k \in V(m)$ . On the other hand, if  $1 - \sum_{l,t:(l,t) \in k^*} \beta_{lt} > 0$ , packet  $m$  is transmitted using the valid schedule  $k^*$ . Our algorithm sets  $X_{mk^*} = 1$ ,  $\alpha_m = 1 - \sum_{l,t:(l,t) \in k^*} \beta_{lt}$ , and updates  $\beta_{lt}$  as  $\beta_{lt} = \beta_{lt}(1 + \frac{1}{C_l}) + \frac{1}{(d_l-1)C_l}$  for all  $(l,t) \in k^*$ , where  $d_l$  is chosen to be  $(1 + 1/C_l)^{RC_l}$ . The complete policy is shown in Algorithm 1.

---

**Algorithm 1** Online Algorithm with Variable  $R$

---

```

1: Initially,  $\alpha_m \leftarrow 0$ ,  $\beta_{lt} \leftarrow 0$ ,  $X_{mk} \leftarrow 0$ .
2:  $d_l \leftarrow (1 + 1/C_l)^{RC_l}, \forall l$ .
3: for each arriving packet  $m$  do
4:    $k^* \leftarrow \operatorname{argmax}_k (1 - \sum_{(l,t) \in k} \beta_{lt})$ 
5:   if  $(1 - \sum_{(l,t) \in k^*} \beta_{lt}) > 0$  then
6:      $\alpha_m \leftarrow (1 - \sum_{(l,t) \in k^*} \beta_{lt})$ 
7:      $\beta_{lt} \leftarrow \beta_{lt}(1 + \frac{1}{C_l}) + \frac{1}{(d_l-1)C_l}, (l,t) \in k^*$ 
8:      $X_{mk^*} \leftarrow 1$ .
9:     Transmit packet  $m$  using valid schedule  $k^*$ .
10:  else
11:    Drop packet  $m$ .
12:  end if
13: end for
```

---

**B. Complexity of the Algorithm**

In step 4, the algorithm finds the valid schedule  $k^*$  that maximizes  $(1 - \sum_{l,t:(l,t) \in k} \beta_{lt})$ . We now show that this step can be completed in polynomial time by dynamic programming. Before presenting the algorithm, some new notations are given as follows. We say that packet  $m$  joins the network at the beginning of time slot  $a_m$ , and specifies its deadline as  $f_m$ . Its source node and destination node are  $s_m$  and  $d_m$ , respectively. Therefore, a valid schedule for  $m$  is one that can deliver a packet from node  $s_m$  to node  $d_m$  between time slots  $a_m$  and  $f_m$ .

Let  $\Theta(n, \tau)$  be the smallest value of  $\sum_{l,t:(l,t) \in k} \beta_{lt}$  among all schedules that can deliver a packet from node  $s_m$  to node  $n$  between time slots  $a_m$  and  $\tau$ .  $\Theta(n, \tau) = \infty$  if there is no schedule that delivers a packet from  $s_m$  to  $d_m$  between time slots  $a_m$  and  $\tau_m$ . Step 4 of Alg. 1 is then equivalent to finding the valid schedule that achieves  $\sum_{l,t:(l,t) \in k} \beta_{lt} = 1 - \Theta(d_m, f_m)$ . Since packet  $m$  arrives at the beginning of time slot  $a_m$ , or, equivalently, at the end of time slot  $a_m - 1$ , we set  $\Theta(s_m, a_m - 1) = 0$  and  $\Theta(n, a_m - 1) = \infty$  for  $\forall n \neq s_m$ .

There are only two different ways to deliver a packet to node  $n$  by the end of time slot  $\tau$ : The first is to

deliver the packet to  $n$  by time slot  $\tau - 1$ , in which case  $\sum_{l,t:(l,t) \in k} \beta_{lt} = \Theta(n, \tau - 1)$ . The second is to deliver the packet to one of  $n$ 's neighbors, say, node  $q$ , by time slot  $\tau - 1$ , and then forward the packet along the link  $l_{qn}$  from  $q$  to  $n$  at time slot  $\tau$ . In this case,  $\sum_{l,t:(l,t) \in k} \beta_{lt} = \Theta(q, \tau - 1) + \beta_{l_{qn}\tau}$ . Therefore, we have

$$\Theta(n, \tau) = \min \begin{cases} \Theta(n, \tau - 1), \\ \Theta(q, \tau - 1) + \beta_{l_{qn}\tau}, q \text{ is a neighbor of } n. \end{cases}$$

Based on the above recursive equation, we design an algorithm for computing  $\Theta(n, \tau)$ . The detailed algorithm is shown in Algorithm 2, where we also use  $Sch(n, \tau)$  to denote the schedule that achieves  $\Theta(n, \tau)$ .

---

**Algorithm 2** Dynamic Programming

---

```

1: for each arriving packet  $m$  do
2:    $\Theta(s_m, a_m - 1) \leftarrow 0$ 
3:    $\Theta(n, a_m - 1) \leftarrow \infty, \forall n \neq s_m$ 
4:    $Sch(n, a_m - 1) \leftarrow \phi, \forall n$ 
5:   for  $\tau = a_m$  to  $f_m$  do
6:     for node  $n$  do
7:        $\Theta(n, \tau) \leftarrow \Theta(n, \tau - 1)$ 
8:        $Sch(n, \tau) \leftarrow Sch(n, \tau - 1)$ 
9:       for node  $n$ 's neighbor  $q$  do
10:        if  $\Theta(q, \tau - 1) + \beta_{l_{qn}\tau} < \Theta(n, \tau)$  then
11:           $\Theta(n, \tau) \leftarrow \Theta(q, \tau - 1) + \beta_{l_{qn}\tau}$ 
12:           $Sch(n, \tau) \leftarrow Sch(q, \tau - 1) \cup \{(l_{qn}, \tau)\}$ 
13:        end if
14:      end for
15:    end for
16:  end for
17: end for
```

---

In Alg. 2, the inequality  $\Theta(q, \tau - 1) + \beta_{l_{qn}\tau} < \Theta(n, \tau)$  is only evaluated once for any link and time slot. Let  $E$  be the number of links in the system. Suppose the number of links is larger than the number of nodes, and  $f_m - a_m + 1 \leq T$ , for all  $m$ , then the complexity of Alg. 2 is  $O(ET)$ .

**C. Competitive Ratio Analysis**

Before analyzing the performance of Algorithm 1, we first establish a basic property of the values of  $\beta_{lt}$ .

**Lemma 2:** Let  $\beta_{lt}[n]$  be the value of  $\beta_{lt}$  after  $n$  packets are scheduled to use link  $l$  at time  $t$ . Then,

$$\beta_{lt}[n] = (\frac{1}{d_l - 1})(d_l^{n/RC_l} - 1). \quad (13)$$

*Proof:* First, note that the value of  $\beta_{lt}$  is only changed when Algorithm 1 uses link  $l$  at time  $t$  to transmit a packet. Therefore, the value of  $\beta_{lt}$  only depends on the number of packets that are scheduled to use link  $l$  at time  $t$ .

We then prove (13) by induction. Initially, when  $n = 0$ ,  $\beta_{lt}[0] = 0 = (\frac{1}{d_l - 1})(d_l^0 - 1)$  and (13) holds.

Suppose (13) holds for the first  $n$  packets. When the  $(n+1)$ -th packet is scheduled for link  $l$  at time  $t$ , we have

$$\begin{aligned}
\beta_{lt}[n+1] &= \beta_{lt}[n](1 + \frac{1}{C_l}) + \frac{1}{(d_l - 1)C_l} \\
&= \frac{1}{(d_l - 1)}(d_l^{n/RC_l} - 1)(1 + \frac{1}{C_l}) + \frac{1}{(d_l - 1)C_l} \\
&= \frac{1}{d_l - 1}[d_l^{n/RC_l}(1 + \frac{1}{C_l}) - 1]
\end{aligned}$$

We select  $d_l = (1 + \frac{1}{C_l})^{RC_l}$ , and therefore

$$\beta_{lt}[n+1] = \frac{1}{(d_l - 1)}[d_l^{(n+1)/RC_l} - 1],$$

and (13) still holds for  $n+1$ . Thus, by induction, (13) holds for all  $n$ . ■

We now establish the competitive ratio of Algorithm 1.

**Theorem 1:** Let  $C_{\min} := \min C_l$ ,  $d_{\min} := (1 + 1/C_{\min})^{RC_{\min}}$ , and  $L$  be the longest path between a source node and a destination node, that is, all valid schedules have  $|k| \leq L$ , for all  $m \in \mathcal{M}$ ,  $k \in V(m)$ . Algorithm 1 produces solutions that satisfy all constraints in **Schedule**( $R$ ) and **Dual**. Moreover, Algorithm 1 is  $(R, 1 + \frac{L}{d_{\min}-1})$ -competitive, which converges to  $(R, 1 + \frac{L}{e^R-1})$ -competitive, as  $C_{\min} \rightarrow \infty$ .

*Proof:* First, we show that the dual solutions  $\{\alpha_m\}$  and  $\{\beta_{lt}\}$  satisfy constraints (10) to (12). Initially, we have  $\beta_{lt} = 0$ . By Lemma 2,  $\beta_{lt} \geq 0$  holds. Since step 6 is only used when  $(1 - \sum_{(l,t) \in k^*} \beta_{lt}) > 0$ ,  $\alpha_m \geq 0$  holds. From step 4 and 6, we know that  $\alpha_m + \sum_{(l,t) \in k} \beta_{lt} \geq (1 - \sum_{(l,t) \in k} \beta_{lt}) + \sum_{(l,t) \in k} \beta_{lt} = 1$ . Thus (10) to (12) hold.

Next, we show  $\{X_{mk}\}$  satisfies constraints (6) to (8). By step 4, the algorithm picks at most one schedule  $k^*$  for packet  $m$ , constraint (6) holds. With Lemma 2,  $\beta_{lt} = 1$  when  $RC_l$  packets use link  $l$  at time  $t$ . Since a valid schedule including  $(l, t)$  will be chosen for packet  $m$  only when  $(1 - \sum_{(l,t) \in k^*} \beta_{lt}) > 0$ , all  $(l, t)$  in the chosen valid schedule must have  $\beta_{lt} < 1$ , and therefore the number of packets transmitted over link  $l$  at time  $t$  must be less than  $RC_l$ . Thus, at any time  $t$ , there are at most  $RC_l$  packets using link  $l$ . Constraint (7) holds. By initialization and step (8), constraint (8) holds.

We derive the ratio between  $\sum_m \alpha_m + \sum_{(l,t)} C_l \beta_{lt}$  and  $\sum_{mk} X_{mk}$ . Initially, both are equal to 0. We consider the increasing amount for both when a new packet  $m$  arrives at the network. We use  $\Delta P(R)$  to denote the change of  $\sum_{mk} X_{mk}$ , and  $\Delta D$  to denote the change of  $\sum_m \alpha_m + \sum_{(l,t)} C_l \beta_{lt}$ .

If packet  $m$  is dropped, both  $\Delta P(R)$  and  $\Delta D$  are 0. If packet  $m$  is accepted and transmitted using valid schedule  $k^*$ , we have  $X_{mk^*} = 1$ . Thus,  $\Delta P(R) = 1$ . On the other hand,  $\Delta D$  is increased as:

$$\begin{aligned}
\Delta D &= \alpha_m + \sum_{(l,t) \in k^*} C_l \Delta \beta_{lt} \\
&= (1 - \sum_{(l,t) \in k^*} \beta_{lt}) + \sum_{(l,t) \in k^*} (\beta_{lt} + \frac{1}{(d_l - 1)C_l}) \\
&= 1 + \sum_{(l,t) \in k^*} \frac{1}{(d_l - 1)} \leq 1 + \frac{L}{d_{\min} - 1}
\end{aligned}$$

Therefore, for each packet arrival, the ratio between  $\Delta D$  and  $\Delta P(R)$  is no larger than  $1 + \frac{L}{d_{\min}-1}$  if  $\Delta D > 0$ . When the algorithm terminates, we have  $\frac{\sum_m \alpha_m + \sum_{(l,t)} C_l \beta_{lt}}{\sum_{mk} X_{mk}} \leq 1 + \frac{L}{d_{\min}-1}$ . By Lemma 1,  $\frac{\Gamma_{opt}}{\sum_{mk} X_{mk}} \leq 1 + \frac{L}{d_{\min}-1}$ , and the competitive ratio of Algorithm 1 is  $(R, 1 + \frac{L}{d_{\min}-1})$ . When  $C_{\min} \rightarrow \infty$ ,  $d_{\min} = (1 + \frac{1}{C_{\min}})^{RC_{\min}} \rightarrow e^R$ , and the competitive ratio of Algorithm 1 converges to  $(R, 1 + \frac{L}{e^R-1})$ . ■

There are several important implications of Theorem 1. First, without increasing capacity, that is, when  $R = 1$ , the competitive ratio of our policy is  $(1, O(L))$ . In comparison, the online algorithm proposed in the recent work [1] focuses on the special case of  $R = 1$  and has a competitive ratio of  $(1, O(L \log L))$ . Therefore, our algorithm is asymptotically better than the online algorithm in [1]. Second, this theorem allows us to quantify the amount of capacity needed to a certain performance guarantee. Suppose the optimal solution to **Schedule** indeed delivers all packets. In order to guarantee that  $1 - \frac{1}{\theta}$  of the packets are transmitted to their destinations before their deadlines, Theorem 1 states that we only need to increase all link capacities by  $R_\theta$  times so that  $1 + \frac{L}{e^{R_\theta}-1} \leq 1/(1 - \frac{1}{\theta}) = 1 + \frac{1}{\theta-1}$ . Therefore, we have  $R_\theta = \ln(L(\theta-1) + 1) \leq \ln L + \ln \theta$ . For example, if we are required to deliver 99% of the packets and the longest path consists of 10 hops, then we need to increase capacity by 6.9 times.

## V. A THEORETICAL LOWER BOUND FOR COMPETITIVE RATIO

In Section IV, we showed that our policy is  $(R, 1 + \frac{L}{e^R-1})$ -competitive. In this section, we will establish a lower bound for the competitive ratio of online policies.

**Theorem 2:** Any online algorithm cannot be better than  $(R, 1 + \frac{L-2e^R}{(L+1)e^R-L})$ -competitive.

*Proof:* We design a network as shown in Fig 2. We start to construct the network from an up-link tree, which is shown as the white nodes in Fig 2. Root is marked as node  $D$  and it is the destination of all packets. There are  $N$  levels of non-root nodes with  $N$  nodes in each level. Each node is connected to one node in the next level. Nodes do not share parent except the  $N$ -th level nodes share the same root node. At the  $j$ -th level, where  $1 \leq j \leq N$ , there are  $\binom{N}{N+1-j}$  extra nodes, which is shown

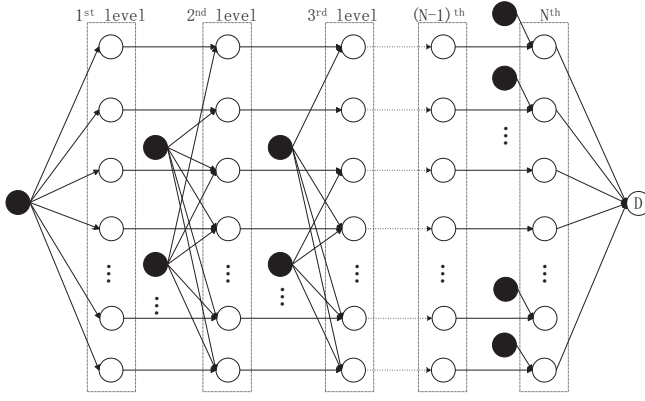


Fig. 2. Network topology for lower bound analysis

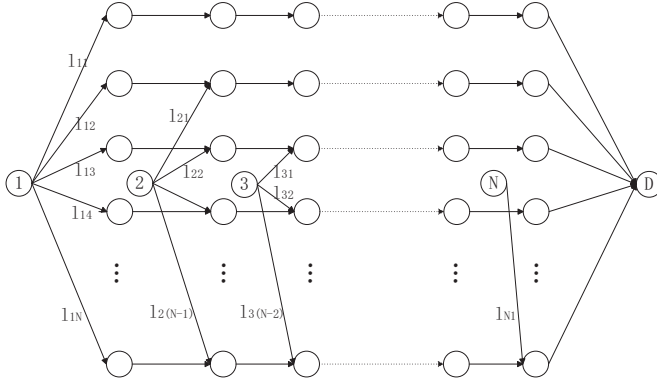


Fig. 3. Simplified network topology for lower bound analysis

as the black nodes in Fig 2, with each node connecting to a unique set of  $N + 1 - j$  nodes in this level. For example, there is one black node connected to all white nodes in level 1, and there are  $N$  black nodes connected to white nodes in level 2, where each of these black nodes is connected all but one white nodes in level 2. Likewise, there are  $\binom{N}{N-2}$  black nodes connected to white nodes in level 3, with each black node connected to  $N - 2$  white nodes in level 3, and no two black nodes are connected to the same subset of white nodes.

Next, we describe packet arrivals. Packets only arrive at black nodes. Of all black nodes connected to the same level of white nodes, only one black node has packet arrival. Let  $\mathcal{W}_j$  be the set of white nodes in  $j$ -th level which connects to the black node with packet arrivals. The black nodes with packet arrivals are chosen such that all nodes in  $\mathcal{W}_{j+1}$  are connected to those in  $\mathcal{W}_j$ . Fig 3 is a simplified network of Fig 2, where we omit the black nodes with no packet arrival and marked each black node with a number from 1 to  $N$ .

Packets arrive at nodes 1, 2, ...,  $N$ . Their destination is

node  $D$ . Each link in the network has capacity  $C$ . At the beginning of time slot 1, there are  $C$  packets arriving at node 1. Node 1 is connected to  $N$  links:  $l_{11}, l_{12}, \dots, l_{1N}$ . At the beginning of time slot 2, there are  $C$  packets arriving at node 2. Node 2 is connected to  $N - 1$  links:  $l_{21}, l_{22}, \dots, l_{2(N-1)}$ . Similarly for nodes 3, 4, ... . At the beginning of time  $N$ , there are  $C$  packets arriving at node  $N$ . The deadline of all packets is  $N + 1$ . Node  $N$  is connected only to link  $l_{N1}$ .

When one knows which black nodes have packet arrivals, the offline optimal algorithm is to transmit the first  $C$  packets through link  $l_{11}$  and the following links, the second  $C$  packets through link  $l_{21}$  and the following links, ..., and the  $N$ -th  $C$  packets through link  $l_{N1}$  and the following link. The total number of delivered packets is  $NC$ .

Next we consider the online algorithm when all links' capacity is increased by  $R$  times. Since online policies do not know which black nodes will have packet arrivals, the optimal online policy is to distribute packets evenly among all connected links. That is, at time 1, each of links  $l_{1i}, i = 1, 2, \dots, N$ , transmit  $C/N$  packets. At time 2, each of link  $l_{2i}, i = 1, 2, \dots, (N - 1)$ , transmits  $C/(N - 1)$  packets. At time  $K$ , link  $l_{Ki}, i = 1, 2, \dots, (N - K + 1)$ , transmits  $C/(N - K + 1)$  packets. For simplicity, we call the routes from node 1 to node  $D$  through  $l_{1i}$  route  $r_i$ . If all packets arrive at node  $K$  are accepted, routes  $r_i, i = K, K + 1, \dots, N$  have the same load on each link. When any link on a single route reaches its capacity, the route cannot be used for future arrival packets. Suppose the route gets over-loaded at time  $K + 1$ , that is, packets arrive at node  $K$  are accepted and packets arrive at node  $K + 1$  are not fully accepted. The maximum load of a single link on route  $r_N$  is at most  $\frac{C}{N} + \frac{C}{N-1} + \dots + \frac{C}{N-K+1}$  and at least  $\frac{C}{N} + \frac{C}{N-1} + \dots + \frac{C}{N-K}$ . We then have:

$$C\left(\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \dots + \frac{1}{N-K+1}\right) \leq RC,$$

and

$$C\left(\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \dots + \frac{1}{N-K}\right) \geq RC.$$

Since

$$\int_{N-K+1}^{N+1} \frac{1}{x} dx < \left(\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \dots + \frac{1}{N-K+1}\right),$$

and

$$\int_{N-K-1}^N \frac{1}{x} dx > \left(\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \dots + \frac{1}{N-K}\right).$$

We have:

$$\log(N + 1) - \log(N - K + 1) = \log \frac{N + 1}{N - K + 1} < R,$$

and

$$\log(N) - \log(N - K - 1) = \log \frac{N}{N - K - 1} > R.$$

Then we can derive the value of  $K$  as:  $N - \frac{N}{e^R} - 1 \leq K \leq N + 1 - \frac{N+1}{e^R}$ . The total number of accepted packets is in the range  $((N - \frac{N}{e^R} - 1)C, (N + 2 - \frac{N+1}{e^R})C)$ .

Thus the competitive ratio of an online policy is at best  $(R, \frac{N}{N+2 - \frac{N+1}{e^R}})$ . In Fig. 2, the longest path in the network is between the leftmost black node and the sink, which has length  $L = N + 1$ . The competitive ratio can then be rewritten as  $(R, 1 + \frac{L-2e^R}{(L+1)e^R-L})$ . ■

Let us once again consider the scenario where online policies need to guarantee to deliver at least  $1 - \frac{1}{\theta}$  as many packets as the optimal solution. Theorem 2 states that any online policy needs to increase its link capacities by at least  $R_\theta$  times so that  $1 + \frac{L-2e^{R_\theta}}{(L+1)e^{R_\theta}-L} \leq 1 + \frac{1}{\theta-1}$ . Solving this equation, and we have  $R_\theta$  needs to be at least  $\ln L + \ln \theta - \ln(L + 2\theta - 1)$ . In comparison, our policy only needs to increase link capacities by  $(\ln L + \ln \theta)$  times to ensure the delivery of  $1 - \frac{1}{\theta}$  as many packets as the optimal solution. Therefore, the capacity requirement of our policy is at most  $\ln(L + 2\theta - 1)$  away from the lower bound. Suppose we fix the ratio between  $L$  and  $\theta$ , and let them both go to infinity, then we have  $(\ln L + \ln \theta) / (\ln L + \ln \theta - \ln(L + 2\theta - 1)) \rightarrow 2$ . Therefore, when both  $L$  and  $\theta$  are large, our policy at most requires twice as much capacity as the theoretical lower bound.

## VI. AN ORDER-OPTIMAL ONLINE POLICY WITH FIXED $R = 1$

We have shown that Alg. 1 is  $(R, 1 + \frac{L}{d_{min}-1})$ -competitive. Without increasing link capacity, i.e.,  $R = 1$ , the algorithm is  $(1, 1 + \frac{L}{e-1})$ -competitive, as  $C_{min} \rightarrow \infty$ . While the competitive ratio of Alg. 1 is an order better than that of the online policy in the previous work [1], it still fails to achieve the theoretical bound of  $(1, O(\log L))$ -competitive. In this section, we propose another online algorithm and prove that it achieves the theoretical bound when  $R = 1$ .

### A. Algorithm Description

Similar to the design of Alg. 1, we aim to design an algorithm that constructs  $\{X_{mk}\}, \{\alpha_m\}, \{\beta_{lt}\}$  while ensuring they satisfy all constraints in **Schedule** and **Dual**. The algorithm is described in Alg. 3. One can see that Alg. 3 is very similar to Alg. 1, and their only difference lie in the update rules for  $\beta_{lt}$ . Specifically, let  $\beta_{lt}[n]$  be the value of  $\beta_{lt}$  when link  $l$  serves a total number of  $n$  packets at time  $t$ . Then Alg. 3 chooses the value of  $\beta_{lt}[n]$  as:

$$\beta_{lt}[n] = \begin{cases} \frac{1}{L(e^{\frac{n}{C_l}} - 1)}, & \text{if } n \leq \frac{C_l}{\ln L+1}; \\ e^{\frac{n}{C_l} - 1}(\ln L+1), & \text{if } n \geq \frac{C_l}{\ln L+1}. \end{cases} \quad (14)$$

To illustrate the difference in  $\beta_{lt}$ , we plot the values of  $\beta_{lt}[n]$  for a link with  $C_l = 1000$  under the two policies in Fig. 4, where we consider the two cases  $L = 8$  and  $L = 64$  for Alg. 3. As can be shown in the figure, when  $n$  is small, Alg. 3 increases the value of  $\beta_{lt}$  much slower than Alg.

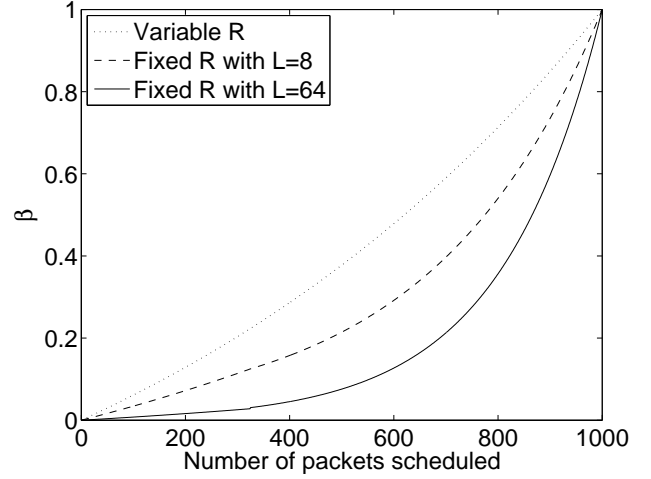


Fig. 4. Values of  $\beta_{lt}$  under different policies.

1 does. Moreover, Alg. 3 increases  $\beta_{lt}$  slower when  $L$  is larger. Recall that both Alg. 1 and Alg. 3 only schedule a packet when  $\max_k (1 - \sum_{(l,t) \in k} \beta_{lt}) > 0$ , or, equivalently,  $\min_k \sum_{(l,t) \in k} \beta_{lt} < 1$ . By increasing  $\beta_{lt}$  slower when  $n$  is small, Alg. 3 ensures that more packets with long routes can be accepted, especially when the network is lightly loaded.

### Algorithm 3 Online Algorithm with Fixed $R = 1$

---

```

1: Initially,  $\alpha_m \leftarrow 0, \beta_{lt} \leftarrow 0, X_{mk} \leftarrow 0$ .
2: for each arriving packet  $m$  do
3:    $k^* \leftarrow \operatorname{argmax}_k (1 - \sum_{(l,t) \in k} \beta_{lt})$ 
4:   if  $(1 - \sum_{(l,t) \in k^*} \beta_{lt}) > 0$  then
5:      $\alpha_m \leftarrow (1 - \sum_{(l,t) \in k^*} \beta_{lt})$ 
6:     for each  $(l, t) \in k^*$  do
7:       if total number of packets  $n$  at time  $t$  on link
          $l: n \leq \frac{C_l}{\ln L+1}$  then
8:          $\beta_{lt} \leftarrow \frac{1}{L(e^{\frac{n}{C_l}} - 1)}(e^{\frac{n}{C_l}} - 1)$ ,
9:       else
10:         $\beta_{lt} \leftarrow e^{\frac{n}{C_l} - 1}(\ln L+1)$ 
11:       end if
12:     end for
13:      $X_{mk^*} \leftarrow 1$ .
14:     Transmit packet  $m$  using valid schedule  $k^*$ .
15:   else
16:     Drop packet  $m$ .
17:   end if
18: end for

```

---

### B. Competitive Ratio Analysis

We now prove that Alg. 3 achieves the theoretical bound in [1] by being  $(1, O(\log L))$ -competitive.

*Lemma 3:* Let  $C_{min} := \min C_l$ . In Algorithm 3, each time a new packet is scheduled, the ratio between the



change of **Schedule** and **Dual** is bounded by  $2(\ln L + 1) + \frac{B}{C_{\min}}$ , where the value of  $B$  is independent of  $C_{\min}$ .

*Proof:* If a new packet is admitted to the network, the increasing amount of **Dual** is

$$\begin{aligned}\Delta D &= \alpha_m + \sum_{(l,t) \in k^*} C_l \Delta \beta_{lt} \\ &= 1 + \sum_{(l,t) \in k^*} (C_l \Delta \beta_{lt} - \beta_{lt})\end{aligned}$$

We define  $\beta(x)$  as

$$\beta(x) = \begin{cases} \frac{1}{L(e^{\frac{1}{\ln L+1}} - 1)}(e^x - 1), & \text{if } x \leq \frac{1}{\ln L+1}; \\ e^{(x-1)(\ln L+1)}, & \text{if } x \geq \frac{1}{\ln L+1}. \end{cases} \quad (15)$$

Note that  $\beta_{lt}[n] = \beta(\frac{n}{C_l})$ . By using Taylor Sequence, we then have

$$\begin{aligned}\Delta \beta_{lt}[n] &:= \beta_{lt}[n+1] - \beta_{lt}[n] = \beta\left(\frac{n+1}{C_l}\right) - \beta\left(\frac{n}{C_l}\right) \\ &\leq \frac{1}{C_l} \beta'\left(\frac{n}{C_l}\right) + \epsilon \frac{1}{C_l^2} \beta''\left(\frac{n}{C_l}\right),\end{aligned}$$

for some bounded constant  $\epsilon < \infty$ , where  $\beta'$  and  $\beta''$  are the first and second derivative of  $\beta$ , respectively. We note that the function  $\beta(x)$  is continuous for all  $x$ , and infinitely differentiable for all  $x$  except at the point  $x_0 := \frac{1}{\ln L+1}$ . At the point  $x_0$ , we define  $\beta'(x_0) = \lim_{x \rightarrow x_0^+} \beta'(x)$  and  $\epsilon \beta''(x_0) = \lim_{x \rightarrow x_0^+} \epsilon \beta''(x)$ . This ensures that the above inequality still holds.

By (14) we know that  $n \leq \frac{C_l}{\ln L+1}$  if and only if  $\beta_{lt}[n] \leq \frac{1}{L}$ .

If  $x = \frac{n}{C_l} \leq \frac{1}{\ln L+1}$ , then  $\beta'(x) = \beta''(x) = \frac{e^x}{L(e^{\frac{1}{\ln L+1}} - 1)}$ .

We have:

$$\begin{aligned}C_l \Delta \beta_{lt}[n] - \beta_{lt}[n] &\leq \frac{C_l \left( \frac{1}{C_l} e^{\frac{n}{C_l}} \right) + \epsilon \left( \frac{1}{C_l} \right)^2 e^{\frac{n}{C_l}} - (e^{\frac{n}{C_l}} - 1)}{L(e^{\frac{1}{\ln L+1}} - 1)} \\ &\leq \frac{1 + \epsilon \frac{1}{C_l} e^{\frac{n}{C_l}}}{L(1 + \frac{1}{\ln L+1} - 1)} \\ &\leq \frac{\ln L + 1}{L} \left( 1 + \epsilon \frac{1}{C_l} e \right)\end{aligned}$$

Let  $B_1 = \epsilon e \frac{\ln L + 1}{L}$ , then

$$C_l \Delta \beta_{lt}[n] - \beta_{lt}[n] \leq \frac{\ln L + 1}{L} + B_1 \frac{1}{C_{\min}}, \quad (16)$$

when  $\frac{n}{C_l} \leq \frac{1}{\ln L+1}$ .

On the other hand, If  $x = \frac{n}{C_l} \geq \frac{1}{\ln L+1}$ , then  $\beta'(x) = (\ln L + 1)\beta(x)$  and  $\beta''(x) = (\ln L + 1)^2 \beta(x)$ . We have:

$$\begin{aligned}C_l \Delta \beta_{lt}[n] - \beta_{lt}[n] &\leq C_l \left[ \frac{\ln L + 1}{C_l} \beta_{lt}[n] + \epsilon \left( \frac{\ln L + 1}{C_l} \right)^2 \beta_{lt}[n] \right] - \beta_{lt}[n] \\ &\leq \ln L \cdot \beta_{lt}[n] + \frac{1}{C_l} \epsilon (\ln L + 1)^2 \beta_{lt}[n]\end{aligned}$$

Let  $B_2 = \epsilon (\ln L + 1)^2$ , then

$$C_l \Delta \beta_{lt}[n] - \beta_{lt}[n] \leq (\ln L + B_2 \frac{1}{C_{\min}}) \beta_{lt}[n], \quad (17)$$

when  $\frac{n}{C_l} \geq \frac{1}{\ln L+1}$ .

If packet  $m$  is transmitted using valid schedule  $k^*$ , we have  $X_{mk^*} = 1$ . Thus,  $\Delta P = 1$ . On the other hand,  $\Delta D$  is increased as:

$$\begin{aligned}\Delta D &= 1 + \sum_{(l,t): (l,t) \in k^*} C_l \Delta \beta_{lt} - \beta_{lt} \\ &\leq 1 + \sum_{(l,t): (l,t) \in k^*, \beta_{lt} \leq \frac{1}{L}} C_l \Delta \beta_{lt} - \beta_{lt} \\ &\quad + \sum_{(l,t): (l,t) \in k^*, \beta_{lt} \geq \frac{1}{L}} C_l \Delta \beta_{lt} - \beta_{lt}\end{aligned}$$

From (16) and (17) we have:

$$\begin{aligned}\Delta D &\leq 1 + \sum_{(l,t): (l,t) \in k^*, \beta_{lt} \leq \frac{1}{L}} \left( \frac{\ln L + 1}{L} + B_1 \frac{1}{C_{\min}} \right) \\ &\quad + \sum_{(l,t): (l,t) \in k^*, \beta_{lt} \geq \frac{1}{L}} ((\ln L + B_2 \frac{1}{C_{\min}}) \beta_{lt})\end{aligned}$$

From Algorithm 3 step 4 we know that  $\sum \beta_{lt} \leq 1$ , thus we have

$$\begin{aligned}\Delta D &\leq 1 + (\ln L + 1 + B_1 \frac{L}{C_{\min}}) + (\ln L + B_2 \frac{1}{C_{\min}}) \\ &= 2 + 2 \ln L + \frac{B_1 + B_2}{C_{\min}},\end{aligned}$$

and the proof is complete.  $\blacksquare$

**Theorem 3:** Algorithm 3 produces solutions that satisfy all constraints in **Schedule** and **Dual**. Moreover, it is  $(1, 2(1 + \ln L))$ -competitive, as  $C_{\min} \rightarrow \infty$ .

*Proof:* First, we show that the dual solutions  $\{\alpha_m\}$  and  $\{\beta_{lt}\}$  satisfy constraints (10) to (12). Initially, we have  $\beta_{lt} = 0$ . By (14),  $\beta_{lt} \geq 0$  holds. Since step 5 is only used when  $(1 - \sum_{(l,t) \in k^*} \beta_{lt}) > 0$ ,  $\alpha_m \geq 0$  holds. From step 3 and 5, we know that  $\alpha_m + \sum_{(l,t) \in k} \beta_{lt} \geq (1 - \sum_{(l,t) \in k} \beta_{lt}) + \sum_{(l,t) \in k} \beta_{lt} = 1$ . Thus (10) to (12) hold.

Next, we show  $\{X_{mk}\}$  satisfies constraints (2) to (4). By step 3, the algorithm picks at most one schedule  $k^*$  for packet  $m$ , constraint (2) holds. With (14), when the number of packets on link  $l$  at  $t$  is  $C_l$ , we have  $\beta_{lt} = 1$ . Also, since a packet is scheduled if  $(1 - \sum_{(l,t) \in k^*} \beta_{lt}) > 0$ , we have  $\beta_{lt} < 1$  for all  $(l, t) \in k^*$ . Therefore, the number

of packets transmitted on link  $l$  at any time  $t$  is at most  $C_l$ . Constraint (3) holds. By initialization and step (13), constraint (4) holds.

When a new packet  $m$  arrives, it will either be dropped or scheduled. If it is dropped, both  $\Delta P$  and  $\Delta D$  are 0. If it is scheduled, both (9) and (1) increase. With Lemma 3, the ratio between  $\Delta P$  and  $\Delta D$  is bounded by  $2(1 + \ln L) + \frac{B}{C_{min}}$ . Therefore the competitive ratio of Algorithm 3 is  $(1, 2(1 + \ln L) + \frac{B}{C_{min}}) \rightarrow (1, 2(1 + \ln L))$ , as  $C_{min} \rightarrow \infty$ . ■

Thus, comparing with the result in [1], Algorithm 3 achieves the optimal competitive ratio when  $R = 1$ .

## VII. A FULLY DISTRIBUTED PROTOCOL FOR IMPLEMENTATION

The two algorithms that we have proposed so far are both centralized algorithms. Specifically, when a packet arrives at a node, the node needs to have complete knowledge of all  $\beta_{lt}$  of all links to find a valid schedule. Such information is usually infeasible to obtain. In this section, we propose a distributed protocol based on the design of Algorithm 1.

In our distributed protocol, the task of transmitting a packet to its destination is decomposed into two parts: First, when a packet arrives at a node, the node determines a suggested schedule based on statistics of past system history. This suggested schedule consists of the route for forwarding the packet, as well as a local deadline for each link. After determining the suggested schedule, the node simply forwards it to the first link of the route. On the other hand, when a link receives a packet along with a suggested schedule, the link tries to forward the packet to the next link in the suggested schedule before its local deadline. The link drops the packet when it cannot forward the packet on time.

To facilitate this protocol, each link keeps track of its own  $\beta_{lt}$ , which reflects the number of packets that are scheduled to be transmitted over link  $l$  at time  $t$ . The value of  $\beta_{lt}$  changes over time, as link  $l$  schedules more and more packets to be transmitted at time  $t$ . Therefore, we define  $\beta_{lt,\hat{t}}$  as the value of  $\beta_{lt}$  when the current time is  $\hat{t}$ . Each link then measures  $\gamma_{l,\tau}$  as the average of  $\beta_{lt,t-\tau}$ . In other words, when the current time is  $t_0$ , the expected value of  $\beta_{lt}$  is  $\gamma_{l,t-t_0}$ . Link  $l$  broadcasts its  $\gamma_{l,\tau}$  periodically so that all nodes can estimate the values of  $\beta_{lt}$ .

We now describe how a node determines a suggested schedule upon the arrival of a packet. Suppose a packet arrives at time  $t_0$ . Following the Alg. 1, the node would like to find a valid schedule that maximizes  $(1 - \sum_{l,t:(l,t) \in k} \beta_{lt})$ . In practice, the node does not know the exact value of  $\beta_{lt}$ . However, it knows that the expected value of  $\beta_{lt}$  is  $\gamma_{l,t-t_0}$ . In our protocol, the node assumes that  $\beta_{lt} = \gamma_{l,t-t_0}$ , and then finds a valid schedule  $k^*$  that maximizes  $(1 - \sum_{(l,t) \in k} \gamma_{l,t-t_0})$ . Similar to Alg. 1, the node drops the packet if  $(1 - \sum_{(l,t) \in k^*} \gamma_{l,t-t_0}) \leq 0$ . If  $(1 - \sum_{(l,t) \in k^*} \gamma_{l,t-t_0}) > 0$ , then the node puts information

of  $k^*$  into the header of the packet, and forwards the packet to the first link in  $k^*$ .

## Algorithm 4 Distributed Implementation: Schedule Suggestion for Each Node

---

```

1: for each arriving packet  $m$  do
2:    $t_0 \leftarrow$  current time
3:    $k^* \leftarrow \operatorname{argmax}_k (1 - \sum_{(l,t) \in k} \gamma_{l,t-t_0})$ 
4:   if  $(1 - \sum_{(l,t) \in k^*} \gamma_{l,t-t_0}) > 0$  then
5:     Put information of the suggested schedule  $k^*$  in
       the header of packet  $m$ .
6:     Forward the packet to the first link in  $k^*$ .
7:   else
8:     Drop packet  $m$ .
9:   end if
10: end for

```

---

Since the actual value of  $\beta_{lt}$  can be different from  $\gamma_{l,t-t_0}$ , there is no guarantee that a packet can be delivered on time using the valid schedule  $k^*$  even if  $(1 - \sum_{(l,t) \in k^*} \gamma_{l,t-t_0}) > 0$ . Therefore, when a node determines a valid schedule  $k^*$  for a packet, the valid schedule  $k^*$  is treated only as a suggestion for links in  $k^*$ . Specifically, if  $k^*$  contains an entry  $(l^*, t^*)$ , then the link  $l^*$  interprets  $k^*$  as a requirement that  $l^*$  needs to forward the packet to the next link before  $t^*$ , or drops the packet. When  $l^*$  obtains the packet, it still has the freedom to choose when to forward the packet, as long as the packet is forwarded before time  $t^*$ .

Next, we discuss how each link determines the actual time to transmit each packet. Obviously, each link  $l^*$  knows its own  $\beta_{l^*t}$ . From the design of Alg. 1, we can see that Alg. 1 prefers to transmit packets when  $\beta_{lt}$  is small. Our proposed policy is based on this principle. When a link  $l^*$  receives a packet, it finds the entry  $(l^*, t^*)$  from the valid schedule  $k^*$  specified in the header of the packet. Link  $l^*$  then finds a time  $t_{tx}$  between the current time and  $t^*$  that has the smallest  $\beta_{l^*t}$ , and transmits the packet at time  $t_{tx}$ . Alg. 5 describes the details of the policy for packet transmission.

## VIII. SIMULATION

In this section, we evaluate the performance of our policies by simulation. We compare our algorithms with EDF policy and the policy, which we call Mao-Koksal-Shroff (MKS) online algorithm, proposed in [1]. Both EDF policy and MKS online algorithm focus on packet scheduling, and are applicable only when the route of the packet is given. For these two policies, we assume that each packet is routed through the shortest path.

We first consider a small network as shown in Fig 5. The network has 9 nodes from node 1 to node 9. There are directed arrows showing the directed links between nodes. All links have the same capacity  $C = 1$ . We assume that there are 1000 packets arriving at the system. For each packet, the source node is chosen uniformly at

---

**Algorithm 5** Distributed Implementation: Packet Transmission for Each Link

---

```

1: for each packet  $m$  do
2:   Upon  $m$ 's arrival at a link  $l^*$ , the link reads schedule
   information  $k^*$  from the header of the packet. Let
    $t^*$  be the local deadline such that  $(l^*, t^*) \in k^*$ .
3:    $t_{tx}^* \leftarrow \operatorname{argmin}_{t_{tx}: t_{tx} \leq t^*} \beta_{l^*, t_{tx}}$ 
4:   if  $\beta_{l^*, t_{tx}^*} < 1$  then
5:      $\beta_{l^*, t_{tx}^*} \leftarrow \beta_{l^*, t_{tx}^*} (1 + \frac{1}{C_{l^*}}) + \frac{1}{(d_{l^*} - 1)C_{l^*}}$ 
6:     Transmit packet  $m$  on link  $l^*$  at time  $t_{tx}^*$ .
7:   else
8:     Drop packet  $m$ .
9:   end if
10: end for

```

---

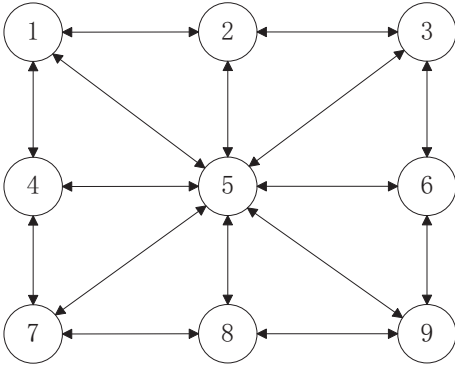


Fig. 5. Network topology for a small network

random between node 1 to node 6, and the destination is chosen uniformly at random between node 7 to node 9. The inter-arrival time between packets are chosen to be 0 with probability 0.7 and 1 with probability 0.3. The deadline of each packet equals its arrival time plus a slack time. The slack time is chosen uniformly from integers between 2 and 6.

Simulation results for different values of  $R$  are shown in Fig. 6. From the result, we can see that all our three policies outperform two other current policies. From the figure, we can see that all our policies are able to deliver all packets when  $R$  is 2. On the other hand, EDF is able to deliver all packets when  $R = 3$ , and MKS can deliver all packets only when  $R$  is as large as 6. We also note that both EDF and MKS are centralized policies. The fact that our distributed algorithm performs better than these two centralized policies further highlights the superiority of our algorithms.

Next, we consider that different links can have different capacities. Since MKS requires all links to have the same capacity, we only compare our policies against EDF. The network topology is also shown in Fig 5. We assume that,

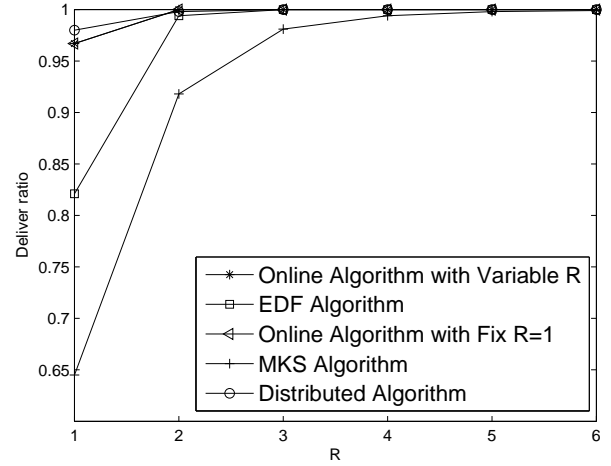


Fig. 6. Deliver ratio comparison when all links have the same capacity.

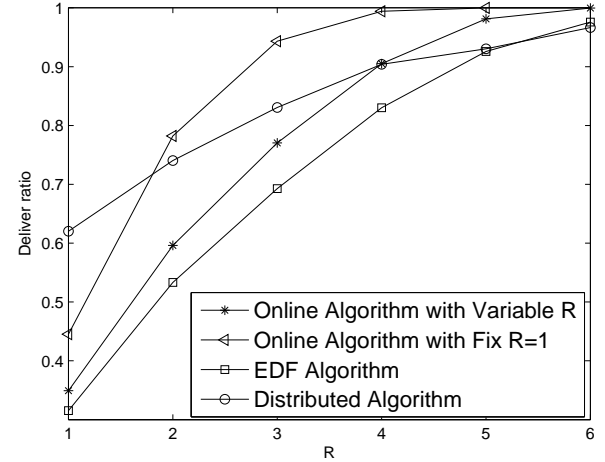


Fig. 7. Deliver ratio comparison when different links have different capacities.

when  $R = 1$ , the link capacity are integers uniformly chosen from 5 to 10. There are 10000 i.i.d packets to be delivered. At the beginning of each time slot, there are a certain number of packets arriving the system. The source node and destination node are both chosen from node 1 to node 9 with equal probability and destination node is not allowed to be the same with source node. The number of packets is randomly chosen between 100 and 500. Each packet has a slack time between arrival and deadline, which is uniformly chosen from  $[2, 6]$ . The result is shown in Fig 7. Once again, we see that our policies, including the distributed algorithm, perform much better than EDF in most cases.

## IX. CONCLUSION

In this paper, we study the multi-hop network scheduling problem with end-to-end deadline and hard transmission rate requirement. Given the capacity of each link in the network, we aim to find out how much capacity we need to increase to guarantee the required ratio of packets can be successfully transmitted to its destination before its deadline without knowing the packet arrival sequences in advance.

We have proposed an online algorithm which works for both fix route and non-fix route network. The algorithm is proved to be  $(R, 1 + \frac{L}{e^R - 1})$ -competitive, where  $L$  is the length of the longest path. We have also showed that the complexity of our algorithm is  $O(ET)$ , where  $E$  is the total number of links and  $T$  is the largest slack time. Next, we have showed that any online algorithm cannot be better than  $(R, 1 + \frac{L - 2e^R}{(L+1)e^R - L})$ -competitive. When both  $L$  and required deliver rate are large, our policy requires at most twice as much capacity as the lower bound. In addition, We have proposed an online algorithm for fixed capacity network. When the capacity cannot be increased, our algorithm is proved to be  $(1, O(\log L))$ -competitive, which is also an order-optimal policy. For practical implementation of our centralized algorithm, we have proposed a heuristic for distributed algorithm so that each node can make decisions without requiring real-time information from all other nodes. In addition to the theoretical results, we compare our policies with two other online policies, including the widely-used EDF policy and a recent proposed policy, by simulation. The results show that the performance of our policies are better than the other two policies. Also the result shows that the distributed algorithm still provide a good delivery ratio.

## X. ACKNOWLEDGMENT

This material is based upon work supported in part by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-15-1-0279 and NPRP Grant 8-1531-2-651 of Qatar National Research Fund (a member of Qatar Foundation).

## REFERENCES

- [1] Z. Mao, C. E. Koksal, and N. B. Shroff, "Optimal online scheduling with arbitrary hard deadlines in multihop communication networks," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 177–189, Feb 2016.
- [2] C. L. Liu and J. W. Layland, "Scheduling algorithms for multi-programming in a hard-real-time environment," *J. ACM*, vol. 20, pp. 46–61, Jan. 1973.
- [3] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang, "On the competitiveness of on-line real-time task scheduling," in *Real-Time Systems Symposium*, 1991. *Proceedings.*, Twelfth, pp. 106–115, Dec 1991.
- [4] S. A. Goldman, J. Parwatikar, and S. Suri, "Online scheduling with hard deadlines," *Journal of Algorithms*, vol. 34, no. 2, pp. 370 – 389, 2000.
- [5] M. H. Goldwasser and B. Kerbikov, "Admission control with immediate notification," *J. of Scheduling*, vol. 6, pp. 269–285, May 2003.
- [6] M. H. Goldwasser, "Patience is a virtue: The effect of slack on competitiveness for admission control," *Journal of Scheduling*, vol. 6, no. 2, pp. 183–211, 2003.
- [7] M. H. Goldwasser and M. Pedigo, "Online nonpreemptive scheduling of equal-length jobs on two identical machines," *ACM Trans. Algorithms*, vol. 5, pp. 2:1–2:18, Dec. 2008.
- [8] J. Ding and G. Zhang, *Online Scheduling with Hard Deadlines on Parallel Machines*, pp. 32–42. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.
- [9] J. Ding, T. Ebenlendr, J. Sgall, and G. Zhang, *Online Scheduling of Equal-Length Jobs on Parallel Machines*, pp. 427–438. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.
- [10] T. Ebenlendr and J. Sgall, "Approximation and online algorithms," ch. A Lower Bound for Scheduling of Unit Jobs with Immediate Decision on Parallel Machines, pp. 43–52, Berlin, Heidelberg: Springer-Verlag, 2009.
- [11] M. Andrews and L. Zhang, "Packet routing with arbitrary end-to-end delay requirements," in *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, (New York, NY, USA), pp. 557–565, ACM, 1999.
- [12] P. P. Bhattacharya, L. Tassioulas, and A. Ephremides, "Optimal scheduling with deadline constraints in tree networks," *IEEE Transactions on Automatic Control*, vol. 42, pp. 1703–1705, Dec 1997.
- [13] V. Rodoplu, S. Vadvalkar, A. A. Gohari, and J. J. Shynk, "Empirical modeling and estimation of end-to-end voip delay over mobile multi-hop wireless networks," in *Global Telecommunications Conference (GLOBECOM 2010)*, 2010 IEEE, pp. 1–6, Dec 2010.
- [14] K. Sanada, N. Komuro, and H. Sekiya, "End-to-end throughput and delay analysis for ieee 802.11 string topology multi-hop network using markov-chain model," in *Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2015 IEEE 26th Annual International Symposium on, pp. 1697–1701, Aug 2015.
- [15] W. Jiao, M. Sheng, K. S. Lui, and Y. Shi, "End-to-end delay distribution analysis for stochastic admission control in multi-hop wireless networks," *IEEE Transactions on Wireless Communications*, vol. 13, pp. 1308–1320, March 2014.
- [16] H. Li, Y. Cheng, C. Zhou, and W. Zhuang, "Minimizing end-to-end delay: A novel routing metric for multi-radio wireless mesh networks," in *INFOCOM 2009, IEEE*, pp. 46–54, April 2009.
- [17] R. Li and A. Eryilmaz, "Scheduling for end-to-end deadline-constrained traffic with reliability requirements in multihop networks," *IEEE/ACM Trans. Netw.*, vol. 20, pp. 1649–1662, Oct. 2012.
- [18] Q. Wang, P. Fan, D. O. Wu, and K. B. Letaief, "End-to-end delay constrained routing and scheduling for wireless sensor networks," in *2011 IEEE International Conference on Communications (ICC)*, pp. 1–5, June 2011.
- [19] I. H. Hou, "Packet scheduling for real-time surveillance in multihop wireless sensor networks with lossy channels," *IEEE Transactions on Wireless Communications*, vol. 14, pp. 1071–1079, Feb 2015.
- [20] R. Singh and P. R. Kumar, "Decentralized throughput maximizing policies for deadline-constrained wireless networks," in *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 3759–3766, Dec 2015.
- [21] X. Liu and L. Ying, "Spatial-temporal routing for supporting end-to-end hard deadlines in multi-hop networks," in *2016 Annual Conference on Information Science and Systems (CISS)*, pp. 262–267, March 2016.